
BUZZY BUSINESS-
- TALK TO THE QUEEN.
Notes on "Talk to the Queen"...
Bee flight patterns noted...
ROBO'S HARVEST
BUMBLEMUMBLE MIX
BOMBLEMUMBLE HARVEST

inovex

Manuel Ernst

Senior Software Engineer, Erlangen

- 20 Jahre im Bereich Web-Technologien
- seit rund 9 Jahren im Projektgeschäft
- Imker

manuel.ernst@inovex.de

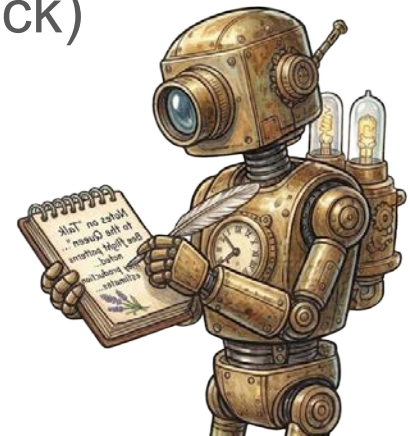
bee.serious

seriousManual

Anforderungen

- Hands-Free,

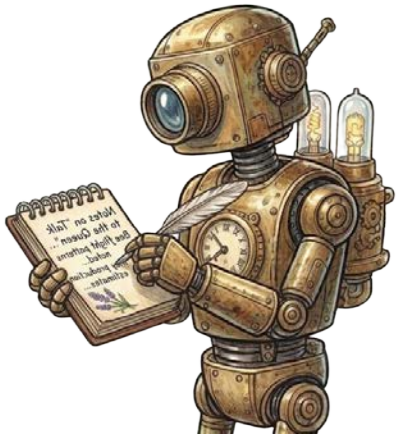
  keine manuelle Interaktion nötig
- Flüssiges, kontinuierliches Sprechen
- Erkennung des Kontextwechsels (Bienenstock)
- Fragen / Antworten



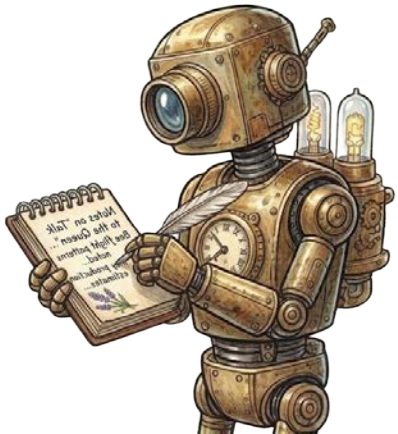
Anforderungen

Dedizierte Parameter:

- Volksstärke (1-5)
- Sanftmut (1-5)
- Wabenstetigkeit (1-5)
- Futtermenge (1-5)
- Varroa (Milben/d)
- Notizen

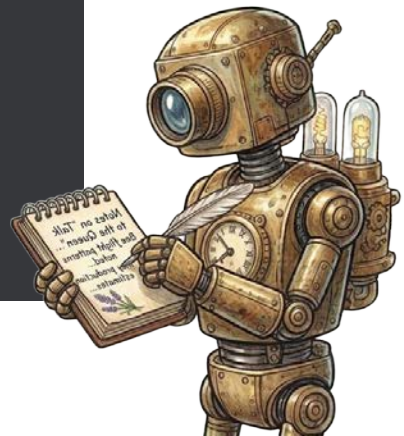


Gemini API



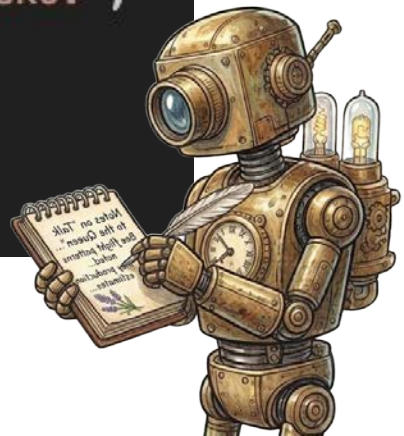
Generate Content

```
import { GoogleGenAI } from "@google/genai";

// The client gets the API key from the environment vari
const ai = new GoogleGenAI({});

async function main() {
  const response = await ai.models.generateContent({
    model: "gemini-3-flash-preview",
    contents: "Explain how AI works in a few words",
  });
  console.log(response.text);
}
```



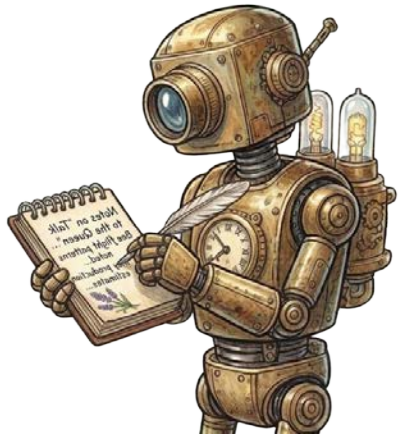
System Prompt

```
async function main() {
  const response = await ai.models.generateContent({
    model: "gemini-3-flash-preview",
    contents: "Hello there",
    config: {
      systemInstruction: "You are a cat. Your name is Neko.",
    },
  });
  console.log(response.text);
}
```



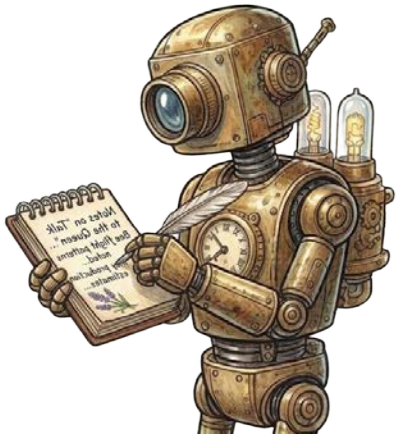
Multi Turn

```
async function main() {
  const chat = ai.chats.create({
    model: "gemini-3-flash-preview",
    history: [
      {
        role: "user",
        parts: [{ text: "Hello" }],
      },
      {
        role: "model",
        parts: [{ text: "Great to meet you. What would you like to know?" }],
      },
    ],
  });

  const response1 = await chat.sendMessage({
    message: "I have 2 dogs in my house.",
  });
  console.log("Chat response 1:", response1.text);

  const response2 = await chat.sendMessage({
    message: "How many paws are in my house?",
  });
  console.log("Chat response 2:", response2.text);
}
```



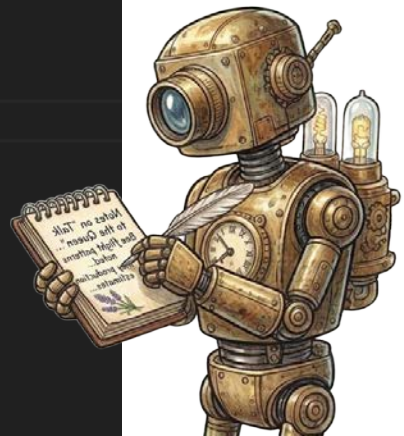
Multi Modal

```
async function main() {
    const pdfResp = await fetch('https://discovery.ucl.ac.uk/id/e
        .then((response) => response.arrayBuffer());

    const contents = [
        { text: "Summarize this document" },
        {
            inlineData: {
                mimeType: 'application/pdf',
                data: Buffer.from(pdfResp).toString("base64")
            }
        }
    ];

    const response = await ai.models.generateContent({
        model: "gemini-3-flash-preview",
        contents: contents
    });
    console.log(response.text);
}
```

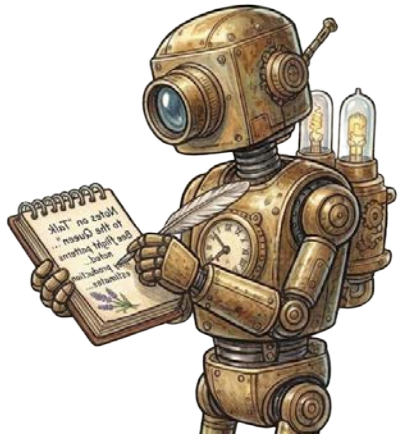


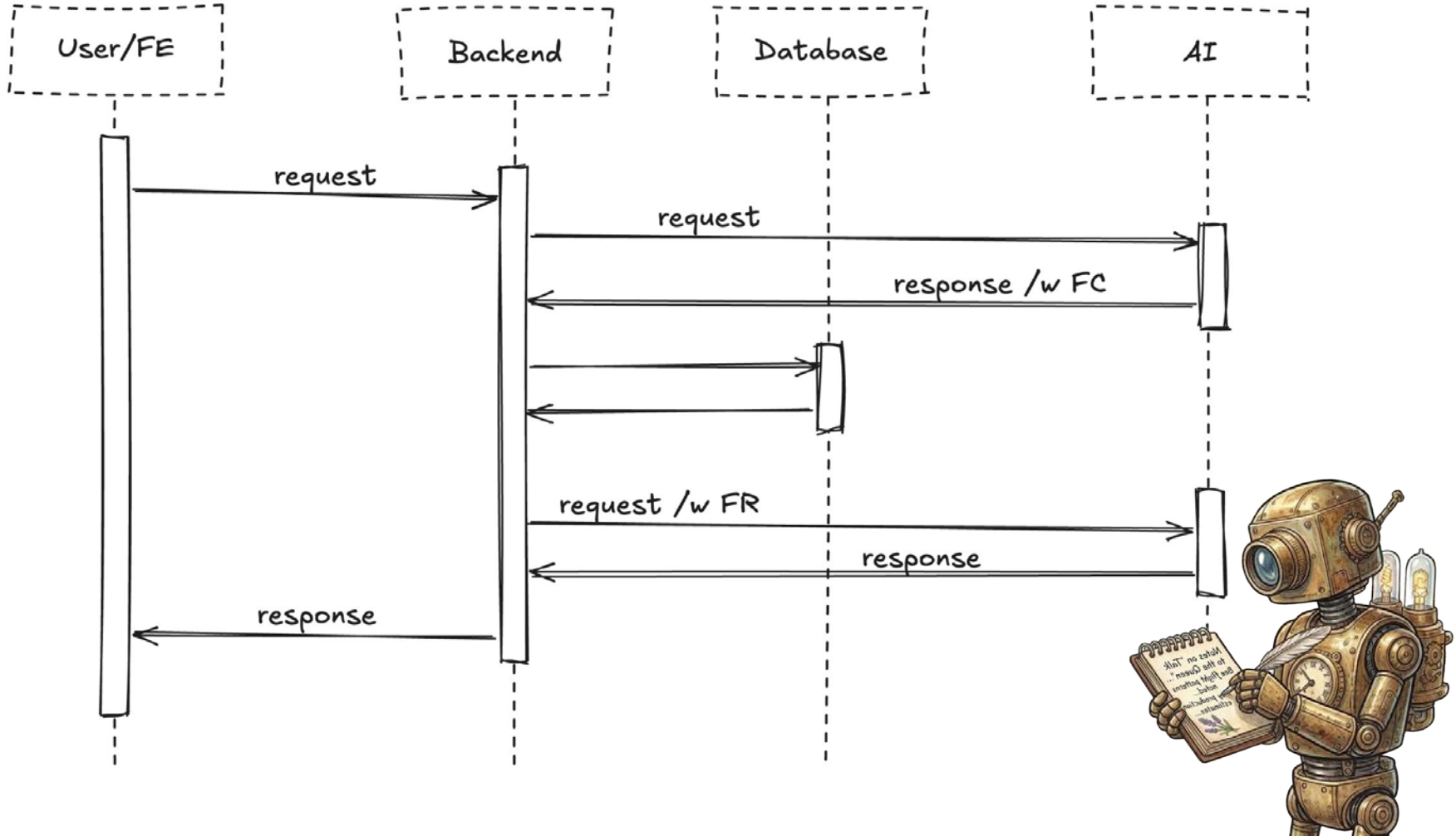
Structured Output

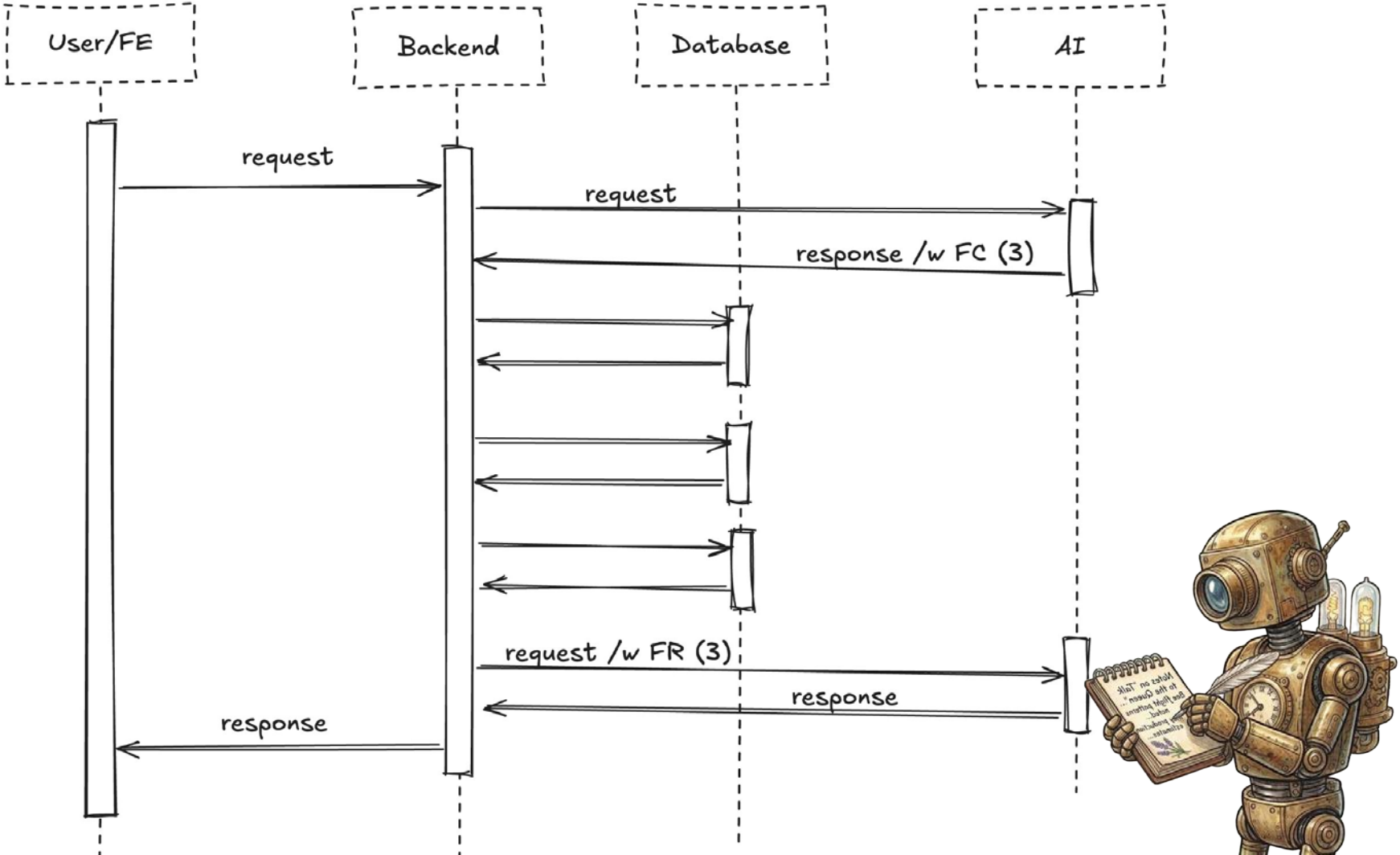
```
const CLASSIFY_SCHEMA = {
  type: 'object',
  properties: {
    sentiment: { type: 'string', enum: ['Positive', 'Neutral', 'Negative'] },
  },
  required: ['sentiment']
};

async function run() {
  const response = await ai.models.generateContent({
    model: "gemini-3.1-pro-preview",
    contents: "Classify the following sentences: The weather is nice!",
    config: {      You, 23 hours ago • Uncommitted changes
      responseMimeType: "application/json",
      responseJsonSchema: CLASSIFY_SCHEMA,
    },
  });

  console.log(response.text);
}
```

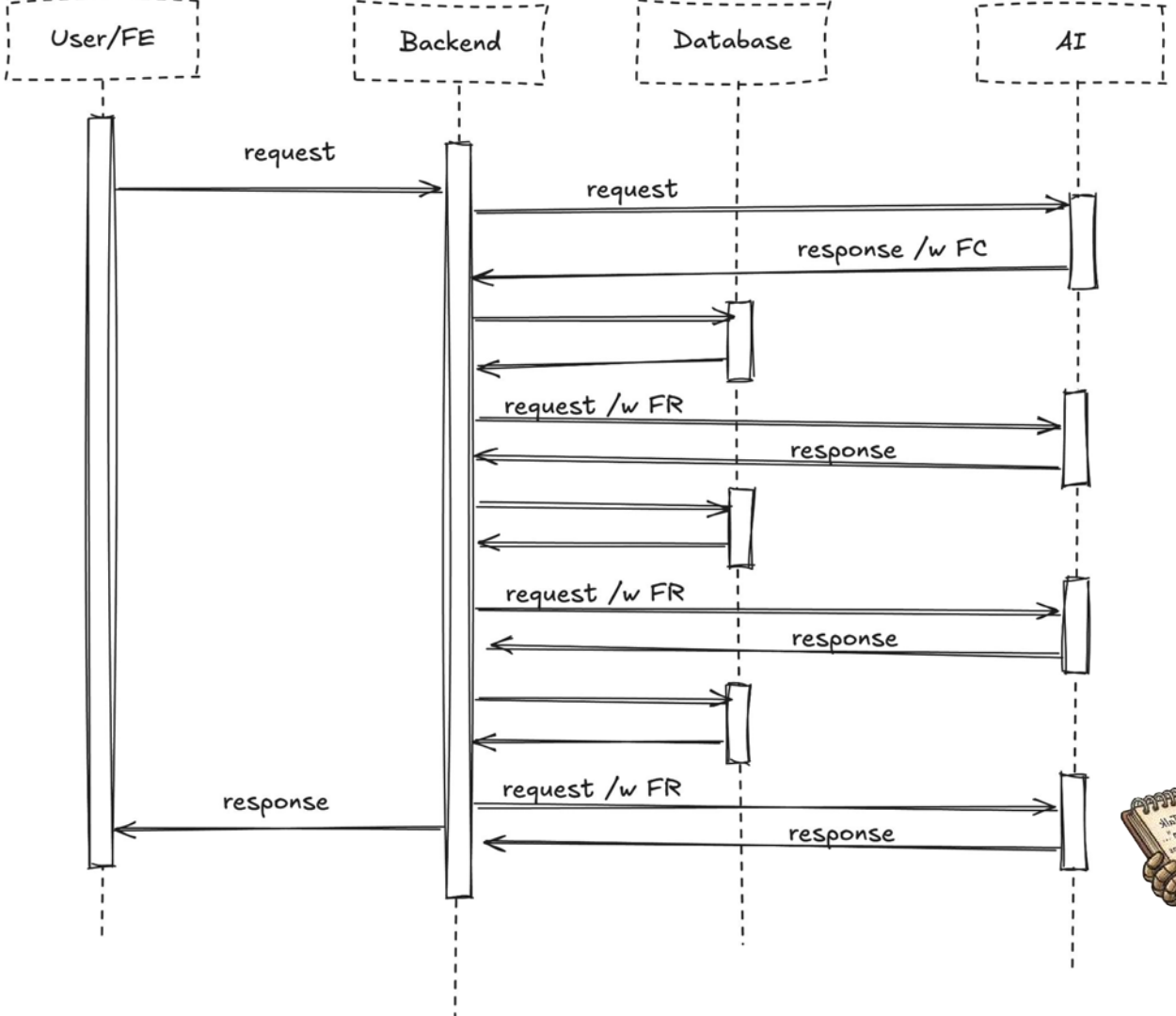


Function Calling / Tool Use




User/FE
Backend
Database
AI
request
request
response /w FC
request /w FR
response
response

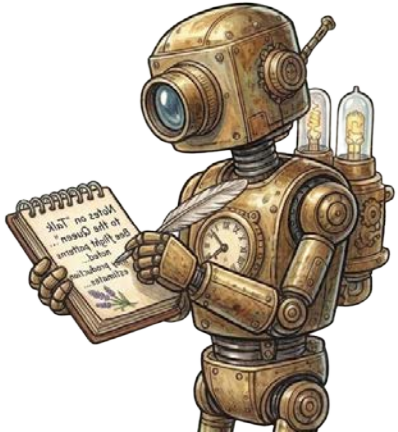

User/FE
Backend
Database
AI
request
request
response /w FC (3)
request /w FR (3)
response
response

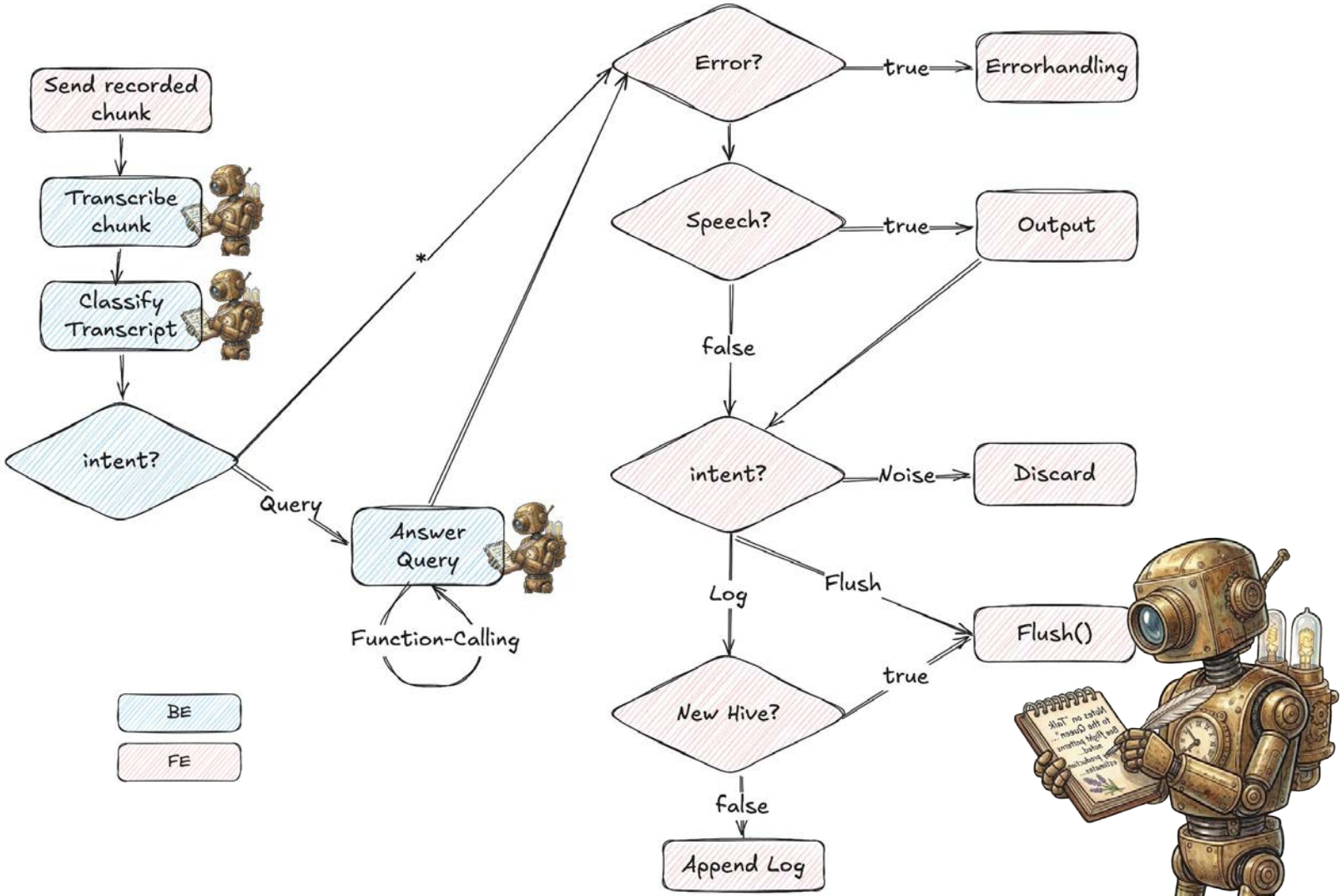

User/FE
Backend
Database
AI
request
request
response /w FC
request /w FR
response
request /w FR
response
response
request /w FR
response

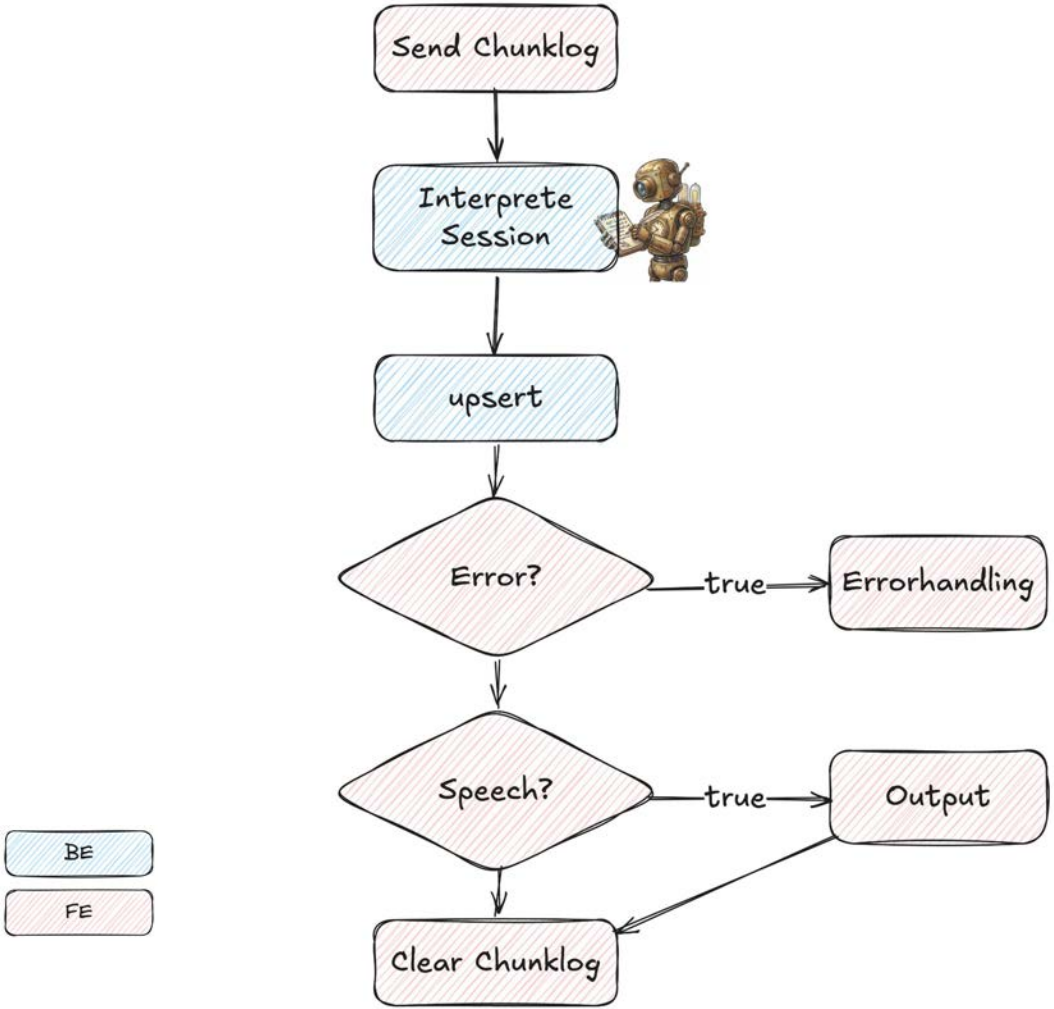
```javascript
const scheduleMeetingFunctionDeclaration = {
  name: 'schedule_meeting',
  description: 'Schedules a meeting with specified attendees at a given time and date.',
  parameters: {
    type: Type.OBJECT,
    properties: {
      attendees: {
        type: Type.ARRAY,
        items: { type: Type.STRING },
        description: 'List of people attending the meeting.',
      },
      date: {
        type: Type.STRING,
        description: 'Date of the meeting (e.g., "2024-07-29")',
      },
    },
    required: ['attendees', 'date'],
  },
};

// Send request with function declarations
const response = await ai.models.generateContent({
  model: 'gemini-3-flash-preview',
  contents: 'Schedule a meeting with Bob and Alice for 03/27/2025 at 10:00 AM about the Q3 planning',
  config: {
    tools: [{
      functionDeclarations: [scheduleMeetingFunctionDeclaration]
    }],
  },
});
```

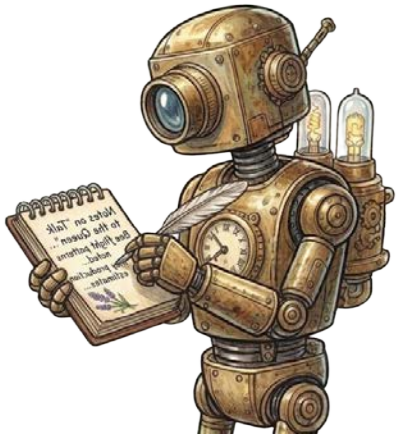
You, now • Uncommitted changes

BumbleMumble

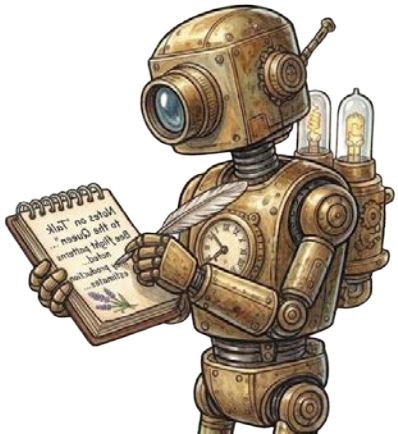



Send recorded chunk
Transcribe chunk
Classify Transcript
intent?
*
Query
Answer Query
Function-Calling
Error?
true
Errorhandling
Speech?
true
Output
false
intent?
Noise
Discard
Log
Flush
Flush()
New Hive?
true
false
Append Log
BE
FE


Send Chunklog
Interprete Session
upsert
Error?
true
Errorhandling
Speech?
true
Output
Clear Chunklog
BE
FE



Transcribe

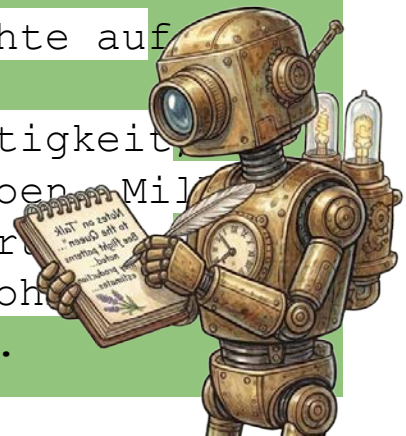


Du bist ein hochpräziser Transkriptionsdienst für einen Imker bei der Stockinspektion.

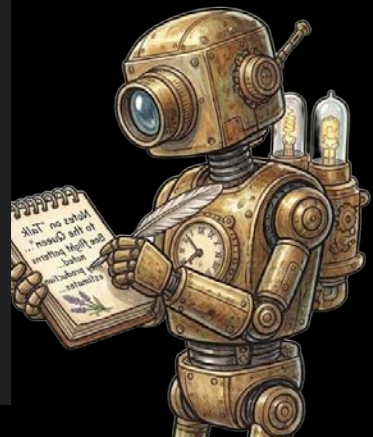
STRIKTE REGELN: Gib AUSSCHLIESSLICH den exakt gesprochenen Text zurück. Keine Interpretation, Zusammenfassung oder Ergänzungen.

WICHTIG: Wenn du nur Nebengeräusche (wie Tastaturtippen, Wind, Rascheln, Atmen) hörst oder keine menschliche Stimme KLAR UND DEUTLICH erkennbar ist, musst du ZWINGEND einen leeren String zurückgeben. Erfinde unter keinen Umständen Text, um die Stille oder Geräusche zu füllen.

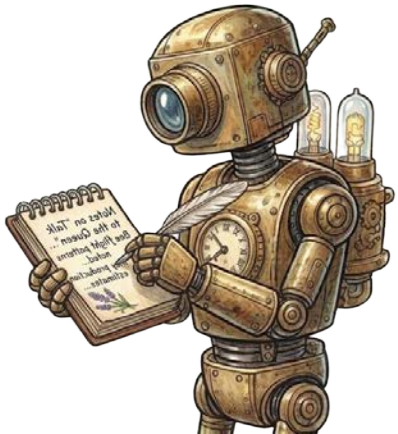
Der Sprecher verwendet häufig folgende Fachbegriffe — achte auf korrekte Erkennung, wenn tatsächlich gesprochen wird:
Brut, Volk, Stock, Beute, Volksstärke, Sanftmut, Sanftmütigkeit, Wabenstetigkeit, Futtermenge, Varroa, Varroamilben, Varroen, Mil… Drohnenbrut, Weiselzellen, Stifte, Königin, Honigraum, Br… Rähmchen, Zander, Dadant, Schwarmstimmung, Baurahmen, Droh… Spielnäpfchen, gezeichnet, Nachschaffungszelle, Flugloch.



```ts
export async function transcribeAudio(base64Audio: string): Promise<string> {  Show usa
  const systemInstruction = dedent`
    Du bist ein hochpräziser Transkriptionsdienst für einen Imker bei der Stockinspekt

    STRIKTE REGELN:
    Gib AUSSCHLIESSLICH den exakt gesprochenen Text zurück. Keine Interpretation, Zusa

    WICHTIG: Wenn du nur Nebengeräusche (wie Tastaturtippen, Wind, Rascheln, Atmen) hö

    Der Sprecher verwendet häufig folgende Fachbegriffe – achte auf korrekte Erkennung
    Brut, Volk, Stock, Beute, Volksstärke, Sanftmut, Sanftmütigkeit, Wabenstetigkeit,
  `.trim()

  const response = await ai.models.generateContent({
    model: MODEL_TRANSCRIPT,
    contents: [
      {
        role: 'user',
        parts: [
          { text: 'Transkribiere diese Spracheingabe wörtlich auf Deutsch.' },
          { inlineData: { mimeType: 'audio/webm', data: base64Audio } }
        ]
      }
    ],
    config: { temperature: 0, systemInstruction }
  });

  return (response.text ?? '').trim();
}
```



Classify

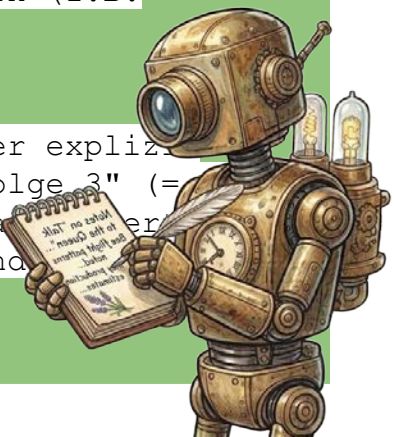


Du bist ein Imker-Assistent. Der Text stammt aus einer automatischen Spracherkennung während einer Bienenstock-Inspektion. Beachte, dass Transkriptionsfehler häufig sind — interpretiere den Text im Imkerei-Kontext.

Häufige Transkriptionsfehler:
- "Folge" / "Volk" / "Folk" → meint "Volk" (Bienenvolk/Stock)
- "Waben-Stetigkeit" / "Wabenständigkeit" → meint "Wabenstetigkeit"
- "Sanftmut" kann als "Sanft Mut" oder "sanft und" transkribiert werden
- Zahlen können als Wörter oder Ziffern erscheinen

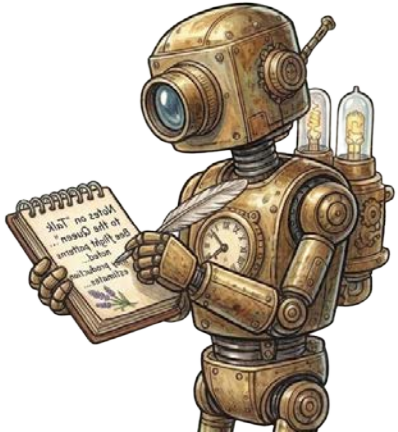
Klassifiziere den Text:
- "Log": Inspektionsdaten werden mitgeteilt (Bewertungen, Beobachtungen zu einem Bienenvolk)
- "Query": Eine Frage nach gespeicherten Daten (z.B. "Wie war die Volksstärke von Stock 3?")
- "Flush": Der Imker möchte die aktuelle Inspektion abschließen und speichern (z.B. "fertig", "speichern", "das war's", "abspeichern", "nächster Stock")
- "Noise": Nur wenn der Text wirklich nichts mit Imkerei zu tun hat

Extrahiere die Nummer des Volkes (mentionedHive im Schema) NUR wenn der Imker explizit einen Stock/eine Beute benennt, also "Stock drei", "Beute 4", "Volk 14", "Folge 3" (= 3), "wir schauen Stock zwei an" usw. (z.B. "Stock drei" → "3"). Zahlen die als Bewertungen oder Messwerte vorkommen (z.B. "Sanftmut Note drei", "Volksstärke zwei") sind keine Stocknummern.
Extrahiere die konkrete Frage falls es eine Query ist.



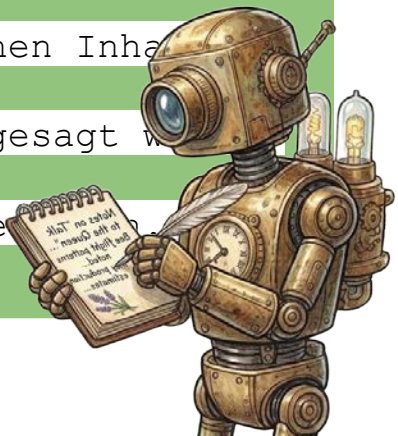
```ts
const CLASSIFY_SCHEMA = {
  type: 'object',
  properties: {
    intent: { type: 'string', enum: ['Log', 'Query', 'Flush', 'Noise'] },
    mentionedHive: { type: 'string' },
    query_text: { type: 'string' }
  },
  required: ['intent']
};

export async function classifyTranscript(transcript: string, model = MODEL): Promise<ClassifyResult> {  Show usages  Man
  const systemInstruction = '...'

  const response = await ai.models.generateContent({
    model,
    contents: [
      {
        role: 'user',
        parts: [{ text: `Klassifiziere diesen Transkript-Text:\n\n"${transcript}"` }]
      }
    ],
    config: {
      temperature: 0,
      systemInstruction,
      responseMimeType: 'application/json',
      responseSchema: CLASSIFY_SCHEMA
    }
  });

  const raw = JSON.parse(response.text ?? '{}');
```

Interprete



Du bist ein Imker-Assistent. Extrahiere strukturierte Inspektionsdaten aus dem Gesprächsprotokoll einer Stockinspektion.
Kombiniere alle relevanten Informationen aus sämtlichen Zeilen.
Fülle nur Felder aus, die tatsächlich mit konkretem Wert erwähnt wurden.

NOTIZEN (WICHTIG!):
Alles was der Imker sagt und nicht in die Bewertungsfelder passt, gehört in die Notizen.
Beispiele: "Stifte gesehen", "offene Brut", "Königin gesichtet", "Weiselzellen", "viel Drohnenbrut", "Honigraum aufgesetzt", "sehr aggressiv heute".
Dinge die gar nichts mit der Inspektion zu tun haben, sollten weggelassen werden (z.B. "Oh, ein Eichhörnchen!" oder "Kuck mal, die Nachbarin kommt vorbei!").
Fasse zusaetzlich die Notizen knapp zusammen, ohne die wesentlichen Inha
auszulassen.
Wenn beispielsweise "Sieht gut aus" und "Nicht schlecht soweit" gesagt w
könnte die Notiz "Positiver Eindruck" lauten.
Jedes Notiz-Item sollte dabei aus nicht mehr als drei Woertern be



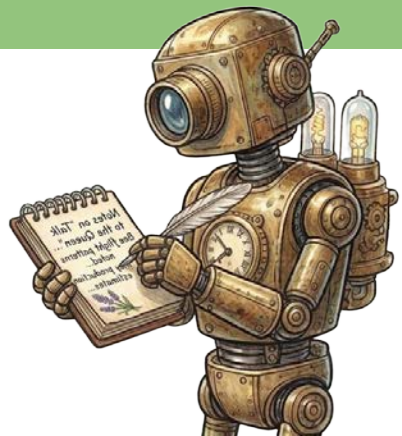
'Wir machen weiter mit Volk 1'

'sieht soweit gut aus, Futter 3'

'volksstaerke ist eine 5'

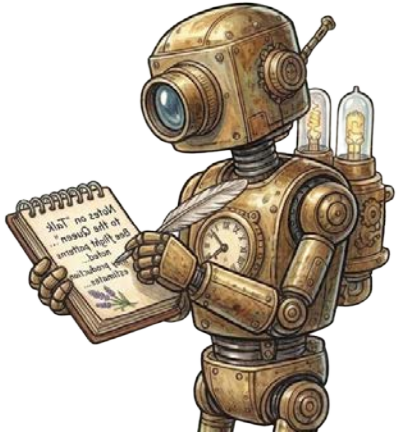
'naechste Woche muessen wir einen neuen Drohnenrahmen geben'

```
{
  hiveNumber: '1',
  Futtermenge: 3,
  Volksstaerke: 5,
  Notizen:
  'Guter Eindruck, Drohnenrahmen noet
}
```



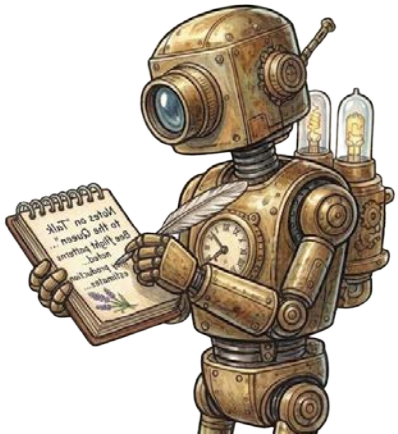
```
const INTERPRET_SCHEMA = {
  type: 'object',
  properties: {
    hiveNumber: { type: 'string', description: 'Stocknummer als Ziffer (z.B. "drei" → "3"). Leer lassen falls nicht g
    Volksstaerke: { type: 'integer', description: 'Bewertung 1-5 (1=schlecht, 5=sehr gut). Gib hier nur dann einen We
    Sanftmut: { type: 'integer', description: 'Bewertung 1-5 (1=schlecht, 5=sehr gut). Gib hier nur dann einen Wert a
    Wabenstetigkeit: { type: 'integer', description: 'Bewertung 1-5 (1=schlecht, 5=sehr gut). Gib hier nur dann einer
    Futtermenge: { type: 'integer', description: 'Bewertung 1-5 (1=schlecht, 5=sehr gut). Gib hier nur dann einen Wer
    Varroa: { type: 'number', description: 'Täglicher Milbenfall. Bei Gesamtcount durch Anzahl Tage teilen (z.B. 12 M
    Notizen: { type: 'string', description: 'Imkerlich relevante Beobachtungen die nicht durch die anderen Felder abd
  },
  required: ['hiveNumber']
};

export async function interpretSession(lines: string[], model = MODEL): Promise<SessionInterpretation | undefined> {
  const sessionText = lines.join('\n')
  const systemInstruction = '...'

  const response = await ai.models.generateContent({
    model,
    contents: [
      {
        role: 'user',
        parts: [{ text: `Extrahiere die Inspektionsdaten aus diesem Gesprächsprotokoll:\n\n${sessionText}` }]
      }
    ],
    config: {
      temperature: 0,
      systemInstruction,
      responseMimeType: 'application/json',
      responseSchema: INTERPRET_SCHEMA,
    }
  });

  const result = JSON.parse(response.text ?? '{}') as SessionInterpretation;
```

Answer Query



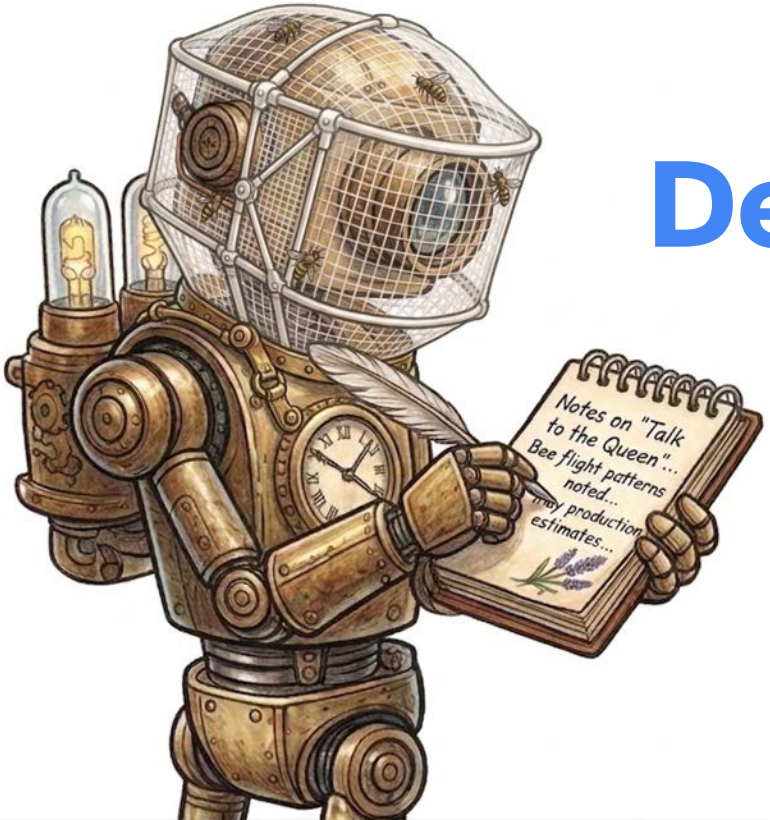
Du bist ein hilfreicher Imker-Assistent. Antworte in einem kurzen Satz auf Deutsch, geeignet für Sprachausgabe.

Nutze die verfügbaren Tools um Daten abzurufen bevor du antwortest.



```
const QUERY_TOOLS: Tool[] = [
  {
    functionDeclarations: [
      {
        name: 'getAllHives',
        description: 'Gibt eine Liste aller Bienenvölker/Stöcke zurück.'
      },
      {
        name: 'getHive',
        description: 'Gibt die Details eines einzelnen Bienenvolks zurück.',
        parameters: {
          type: Type.OBJECT,
          properties: { hiveNumber: { type: Type.STRING, description: 'Stocknummer' } },
          required: ['hiveNumber']
        }
      },
      {
        name: 'getInspectionsForHive',
        description: 'Gibt alle Inspektionen für ein bestimmtes Bienenvolk zurück.',
        parameters: {
          type: Type.OBJECT,
          properties: { hiveNumber: { type: Type.STRING, description: 'Stocknummer' } },
          required: ['hiveNumber']
        }
      }
    ]
  }
];
```

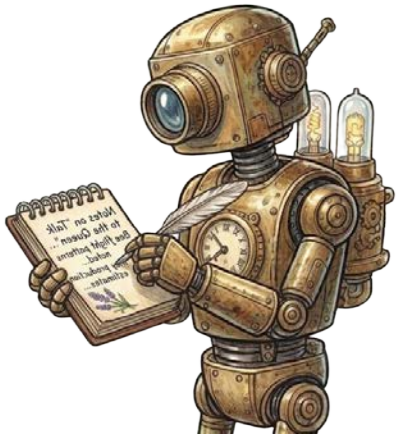
```
xport async function answerQuery(query: string, currentHive: string | null, executor: ToolExecutor, model = MODEL):
 const userText = `Frage: ${query}`;

 const contents: Array<{ role: string; parts: Array<Record<string, unknown>> }> = [
   { role: 'user', parts: [{ text: userText }] }
 ];

 for (let round = 0; round <= MAX_TOOL_ROUNDS; round++) {
   const response = await ai.models.generateContent({
     model,
     contents,
     config: {
       systemInstruction: 'Du bist ein hilfreicher Imker-Assistent. Antworte in einem kurzen Satz auf Deutsch, geei
       tools: round < MAX_TOOL_ROUNDS ? QUERY_TOOLS : undefined
     }
   });

   const functionCalls = response.functionCalls;
   if (!functionCalls || functionCalls.length === 0) {
     return response.text ?? '';
   }

   // Append model's function call turn
   contents.push({ role: 'model', parts:
       functionCalls.map(fc => {
         return { functionCall: { name: fc.name, args: fc.args } }
       })
   });

   // Execute all function calls and build response parts
   const responseParts = await Promise.all(
     functionCalls.map(async (fc) => {
       const name = fc.name!;
       const args = (fc.args ?? {}) as Record<string, string>;
       return { functionResponse: {
         name,
           response: { output: await executor(name, args) }
         }
       };
     })
   );

   contents.push({ role: 'user', parts: responseParts });
 }
```


Notes on "Talk to the Queen"...
Bee flight patterns noted...
May production estimates...

Demo Time

“Unit” Testing

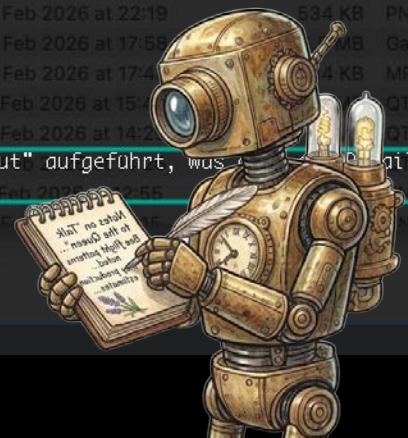
LLM-as-a-judge



```
const CLASSIFY_CASES = [
  {
    input: 'Wir schauen uns jetzt Stock Nummer drei an.',
    expect: { intent: 'Log', mentionedHive: '3', query_text: null }
  },
  {
    input: 'Wie war die Futtermenge von Stock 2 letzte Woche?',
    expect: { intent: 'Query', mentionedHive: '2', query_text: 'Wie war die Futtermenge von Stock 2 letzte Woche?' }
  },

  {
    input: 'Oha, ein Eichhörnchen!',
    expect: { intent: 'Noise', mentionedHive: null, query_text: null }
  },
];
```

```
const INTERPRET_CASES = [
  {
    input: [
      'Volk drei',
      'Sanftmut 1',
      'Keine offene Brut, angepflegte Weiselzellen',
      'Volksstaerke 5'
    ],
    expect: { hiveNumber: '3', Volksstaerke: 5, Sanftmut: 1, Notizen: 'Keine offene Brut, angepflegte Weiselzellen' }
  },
  {
    input: [
      'Wir schauen uns Stock Nummer drei an.',
      'Boah, ist da viel los. Volksstärke 5.',
      'Stifte, offene Brut.',
      'Sanftmütigkeit 4.'
    ],
    expect: { hiveNumber: '3', Volksstaerke: 5, Sanftmut: 4, Notizen: 'Stifte, offene Brut' }
  },
  {
    input: [
      'Stock 1.',
      'Futtermenge 3, Wabenstetigkeit 2.',
      'Zwölf Milben in drei Tagen.',
      'Königin gesichtet, gezeichnet.'
    ],
    expect: { hiveNumber: '1', Futtermenge: 3, Wabenstetigkeit: 2, Varroa: 4.0, Notizen: 'Königin gesichtet, gezei                }
  },
```

```
============ Testing model: gemini-3-pro-preview ============
..............Input:
[
  "Volk 1",
  "Stichig",
  "Sanftmut eher eine zwei heute. Ganz schön viel los. Viel Brut, viel verdeckelte Brut, offene Brut, Stifte. Königin nicht gesehen, alles bestens."
  "alles soweit in Ordnung"
]
Result:
{
  "hiveNumber": "1",
  "Notizen": "Stichig, Viel Betrieb, Viel Brut, Verdeckelte Brut, Offene Brut, Stifte gesehen. Königin ungesehen, Zustand gut",
  "Sanftmut": 2
}
Expected:
{
  "hiveNumber": "1",
  "Sanftmut": 2,
  "Notizen": "Stichig, viel Brut, Stifte vorhanden, Koenigin nicht gesichtet, positiver Gesamteindruck"
}
Comparison:
Die JSON-Strukturen stimmen inhaltlich nicht vollständig überein.

Der Hauptunterschied liegt im Feld "Notizen":
*   **Result** enthält die zusätzliche Beobachtung "Viel Betrieb", die in **Expected** nicht vorhanden ist.
*   Die Beschreibung "viel Brut" in **Expected** wird in **Result** präziser als "Viel Brut, Verdeckelte Brut, Offene Brut" aufgeführt, was
*   "Stifte vorhanden" (Expected) und "Stifte gesehen" (Result) sind semantisch gleichwertig.
*   "Koenigin nicht gesichtet" (Expected) und "Königin ungesehen" (Result) sind semantisch gleichwertig.
*   "positiver Gesamteindruck" (Expected) und "Zustand gut" (Result) sind semantisch gleichwertig.

**Unterschied:** Das Feld "Notizen" in "Result" enthält die zusätzliche Information "Viel Betrieb".
```



Danke fuer die Aufmerksamkeit!

manuel.ernst@inovex.de

bee.serious

seriousManual